Chapter 5

Continuous Descriptions

While a particle system representation of shape is sufficient for many applications, continuous surface descriptions are the standard method of describing shape in computer graphics. This chapter focuses on generating continuous surface descriptions from particle systems. First, we discuss methods used to generate surface descriptions from volumetric particle systems. Second, we discuss methods used to generate surface descriptions from particle systems of surface elements. These are different problems requiring different solutions.

5.1 Surface Descriptions Based on Volume Samplings

5.1.1 Implicit Representation

For volume based particle samplings, an *implicit* formulation of the surface provides a compact mathematical description. An implicit surface is defined as the locus of points that obey a point classification function, such as $f(x, y, z) = 0$. Assuming there exists a scalar field function that varies throughout space, an *iso-surface* is the locus of points in space whose scalar field value equals a given constant. When the field function varies continuously in space and without discontinuities, the function defines a set of closed continuous 3D surfaces. An iso-surface of threshold T is easily defined by an implicit function f as

$$f(x, y, z) = F(x, y, z) - T = 0,$$

where $F(x, y, z)$ is the value of the field in space.

To define an iso-surface equation for a particle system, we assign a continuous field to each particle. The field is maximum at the particle's center and monotonically decreases as a function of distance from the particle. The surface comprises all points in space for which the sum of the particle fields equals a threshold constant. Formally we write the iso-surface equation f as

$$f(x, y, z) = \sum_i g_i(x, y, z) - T = 0, \tag{5.1}$$

where g_i is the field function for particle i and T is the threshold value.

Blinn (1982) introduced such a class of algebraic surfaces based on control points and exponential field functions. The exponential field function results in smooth surfaces, but because the individual fields extend to infinity, it is expensive to compute the field for a given point in space. To reduce the computational effort, bounded polynomial functions of a similar shape have been used instead (Wyvill, McPheeters and Wyvill, 1986b). However, to maintain C^1 or C^2 surface continuity, one must be careful when choosing the polynomial to use.

5.1.2 Explicit Representation

By sampling the field function in space we can generate a C^0 continuous *explicit* description of the surface. The description is a polygonal approximation of the implicit surface representation, with the benefit of being easily imported into almost all commercial and public domain software rendering packages. Most iso-surface polygonization techniques are based on four steps. First, sampling points in space. Second, categorizing the points as inside or outside of the surface. Third, determining the surface intersection points between pairs of adjacent inside/outside points. And fourth, fitting polygons to the surface intersection points.

The well known Marching Cubes algorithm of (Lorensen and Cline, 1987) samples space on a cubic grid and polygonizes each cell independently. Unfortunately, ambiguous polygonizations occur since more than one possible plane will match certain combinations of in/out vertices for a given cube. These cases can be resolved by testing additional points (Wyvill, McPheeters and Wyvill, 1986b), by applying surface coherence between adjacent cells (Baker, 1988), or by sampling and testing the vertices of a tetrahedron instead of a cube (Bloomenthal, 1988; Velho, 1990).

Higher resolution polygonal approximations come at the cost of a finer grid sampling. The majority of these samples fall in empty space and provide no direct benefit. This can be observed by looking at what happens when we double the sampling rate along each axis. The number of samples in space grows by $O(N^3)$ while the number of polygons grows at only $O(N^2)$. Adaptive polygonizations benefit from concentrating polygons in areas of high curvature while reducing the number of polygons in areas of low curvature to produce a more accurate approximation for a limited number of polygons (Bloomenthal, 1988; Velho, 1990; Hall and Warren, 1990). Alternatively, methods that spread across the surface from a known surface point minimize cost by restricting computation to sample points near the surface.

5.1.3 Direct Surface Sampling

If specific points on the iso-surface are required, one can directly sample the surface by computing ray-surface intersections. The surface points are found by combining the ray equation with the implicit surface equation (5.1) and solving for the roots. The problem is complicated by the fact that for a system of N particles, in the worst case, there may be as many as $2N$ intersections with a given ray and thus $2N$ roots to the equation. For example, imagine all of the particles lying on the X axis

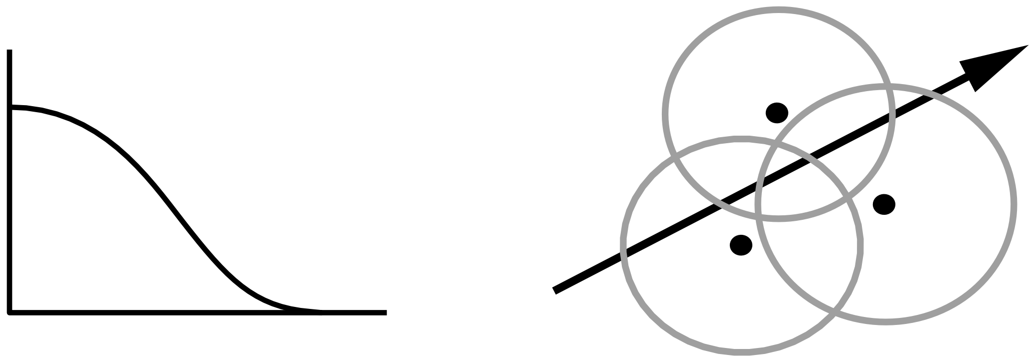


Figure 5.1: Ray tracing particle iso-surfaces

(a) A particle field function, defined as a scalar algebraic function of distance from the particle. (b) Ray intersecting field function bounding spheres. In this example the ray is split into seven non-overlapping intervals.

and spaced such that the resulting iso-surface of the field is N distinct spheres. A more likely case, and one that the potential energies encourage, is the clustering of particles into volumes, with each particle approximately r_o distance from its nearest neighbors. In this case a ray will pass through $N^{\frac{1}{3}}$ of the particle fields on average. If only the first intersection of the ray with the surface is needed, such as for rendering opaque surfaces, the problem becomes less complex. Assuming each particle field function is monotonically decreasing, one can guess an interval along the ray where the first intersection will occur (Blinn, 1982). The intersection is then isolated using an iterative root solver.

For particle fields restricted to polynomials of bounded range, there exists an algorithm to find *all* of the intersections of the iso-surface with a ray in time linear in the number of particles (Tonnesen, 1989; Wyvill and Trotman, 1989). The key idea is to split the ray into non-overlapping intervals such that each interval is represented by a single continuous algebraic equation. Figure 5.1(a) shows the graph of particle's field function as a function of distance from the particle position. A scalar field F in R^3 is defined as the summation of individual particle field functions. Since the individual field functions decay to zero at a fixed distance each separate particle field is bounded by a sphere. Figure 5.1(b) shows a simple 2D example of particles, their bounding circles, and a ray. In the 3D case, the iso-surface must lie within the union of all bounding spheres. The intersection of the bounding spheres partition space into regions, such that in each region the scalar field F is defined by the summation of a subset of the algebraic equations defining the particle fields. The intersection of a ray and the bounding spheres split the ray into non-overlapping intervals. Solving for the roots of the combined ray-field equation, over the interval, yields the intersection points of the ray and surface within that interval. For low order polynomials, computing the analytical solution is considerably faster than an iterative root finding approach. Another benefit of this approach is that it is suitable for constructive solid geometry modeling systems, which require all ray/surface intersections.

5.2 Surface Descriptions Based on Surface Samplings

We now consider the problem of generating surface descriptions from particle systems of surface samples. This is the *surface reconstruction problem.* In particular we want the surface function to *interpolate* the data, passing through the data points. This is in contrast to a surface function which approximates the data, passing close to but not necessarily through the data points. The latter functions are appropriate when the data points may not necessarily be lying on the surface, such as when there is noise in the sampling process, and the desire is to hide the noise through a smoothing process.

Reconstruction methods are classified as either *global* or *local* in nature. In a global technique, any given surface patch is dependent upon all of the data. In a local technique, any given surface patch is only dependent on nearby data points. In highly structured data, such as data sampled over a grid, the neighborhood relationships are known a priori. In our case, the case of unstructured point data, we must determine the neighboring particles. Franke (1982) observes if the data are scattered, one must inspect, in some way, all of the data to determine which points are nearby. The question arises as to whether there is such a thing as a "local" method for scattered data. Since the nature of a local surface fitting or interpolation function does not depend on how the neighbors are found, only that they are found, we may consider this to be a rhetorical question. However it reminds us that nearest neighbors must be found. The calculation of nearest neighbors is discussed in Chapter 7.2.

We have designed our particle systems so that areas of surface will be represented by collections of evenly spaced sheets of particles. Our first reconstruction goal is to reconstruct a surface which interpolates the particle positions. Our second reconstruction goal is to limit surface reconstruction to areas where the particles are sufficiently close, where closeness is defined by a distance measure. In other areas, where particles are sufficiently far apart, the goal is for the reconstructed surface to be discontinuous, with breaks or holes.

Surface reconstruction is simplified from the general case if the structure of the original surface is known. By surface structure we mean the geometric relationship between points on the surface. For example, if a set of position or point data maps uniquely to a parameterized surface such as a plane or sphere, that surface imposes a global structure over the data, and the reconstruction problem reduces to finding the best local structure that matches the global structure.

The reconstruction problem is much harder when the topology and genus is unknown, as in our case. Neither are we assuming an open or closed or surface, but which ever matches the given surface samples the "best". Since we cannot assume a global structure, we must instead be able to derive the structure from properties intrinsic to the data. One approach is to use divide and conquer, solving for local structure everywhere over subsets of the data and then combining the results to create the global structure. By choosing sufficiently small regions of shape with respect to the curvature, we can assume a locally planar structure thereby reducing the sub-

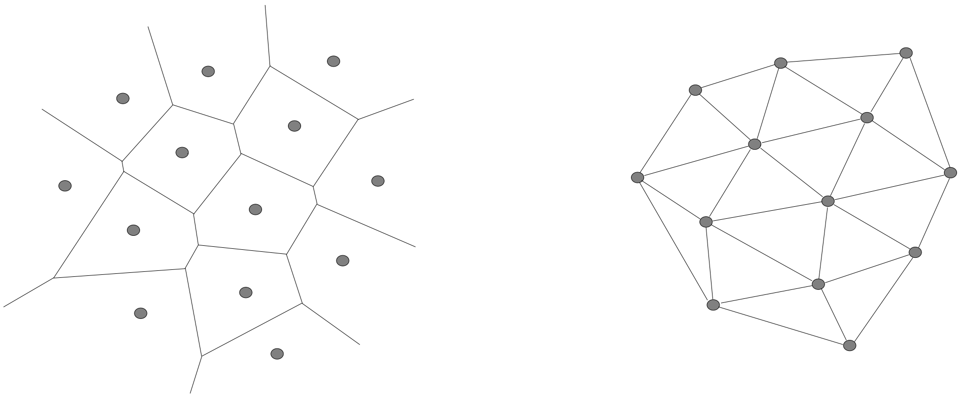


Figure 5.2: Voronoi diagram and Delaunay triangulation

(a) Voronoi diagram in 2D. (b) Delaunay graph (triangulation) as the straight line dual of the Voronoi diagram.

problem to the two dimensional case. We start by considering structure over localized areas of the shape.

5.2.1 Local Structure

An obvious property of a continuous surface is that points over a small patch are in close spatial proximity to each other. Thus it is natural to define structure based on the spatial proximity relationship between points. We consider the two dimensional problem first. The Voronoi diagram[1] can be used to solve a number of proximity problems. In particular, it solves the "loci of proximity problem". For two dimensions this is defined as (Preparata and Shamos, 1985):

> Given a set S of N points in the plane, for each point p_i in S, find the locus of points (x, y) in the plane that are closer to p_i than to any other point of S.

The solution is a partitioning of the plane into regions of spatial proximity to the original data points (Figure 5.2a). Computing the Voronoi diagram requires $O(N \log N)$ optimal computation time.

While the Voronoi diagram provides a structure over the data, it is not suitable as a surface description, since it contains unbounded edges in 2D. A better solution is to define a triangulation over the points which encodes the same proximity information. Such a triangulation is the *Delaunay triangulation*, the dual of the Voronoi diagram. A Delaunay triangulation is the graph embedded in the plane obtained by adding a straight-line segment between each pair of points whose Voronoi polygons share

[1]Voronoi diagrams also go by the names Dirichlet regions, Thiessen polygons, Wigner-Seitz cells, and proximal polygons.

an edge[2]. It is also defined as the unique triangulation such that the circumcircle[3] of each triangle does not contain any other point of S in its interior. Figure 5.2(b) shows such a triangulation constructed from the Voronoi diagram in Figure 5.2(a). The Delaunay triangulation can be computed from the Voronoi diagram in $O(N)$ time, or directly in $O(N \log N)$ optimal time. Computationally, both approaches are equivalent (Preparata and Shamos, 1985, pg. 217).

Other suggested criteria for producing good two dimensional triangulations are:

- maximizing the minimum interior angle,
- minimizing the roughness measure over a height field, and
- minimizing the total length of edges.

Intuitively, a good triangulation should avoid long thin triangles in favor of triangles that are "more or less equilateral". More formally, the criterion is to *maximize the minimum interior angle* of the triangles over all possible triangulations (Lawson, 1977). Lawson goes on to show that the max-min angle criterion, the circle criterion, and the straight-line dual of the Voronoi diagram produce equivalent triangulations in the plane.

Related to 2D triangulation is the triangulation of height field data. Rippa (1990) has suggested that a good triangulation of height field data is one that corresponds to the intuitive concept of the "smoothest surface". He defines a roughness measure as the L^2 norm squared of the gradient of the triangulation. If g_i is the planar interpolating surface for a triangle T_i then the roughness measure of T_i is

$$\int_{T_i} \left[\left(\frac{\partial g_i}{\partial x}^2 \right) + \left(\frac{\partial g_i}{\partial y}^2 \right) \right] \mathrm{d}x \ \mathrm{d}y,$$

and the roughness measure of the entire triangulation is the sum of the roughness measures of all triangles. He then goes on to show that *minimizing the roughness measure* of the triangulated height field is equivalent to the Delaunay triangulation in the plane. It is interesting to note that while the triangulation minimizes the roughness, it is also independent of values defining the height of the surface.

Another criterion for 2D triangulation is to select triangles of short edge length over longer edges. The *minimum-weight triangulation* is the triangulation that exhibits the minimal total length over all triangulation edges. At one time it was conjectured that the Delaunay triangulation was a minimum-weight triangulation, until this was disproved by Lloyd (Preparata and Shamos, 1985). Our particle systems do not present the general case of random point samples, because they naturally arrange into hexagonal configurations of nearly equal spacing and nearly equal angles between neighboring points. Thus for our systems, we expect the minimum weight and the Delaunay algorithms to generate similar if not identical triangulations.

[2] As stated this is true when there are not four or more co-circular points. However, the triangulation of these cases is straightforward.

[3] The circumcircle of a triangle is the circle such that each vertex is on the perimeter of the circle. There is exactly one circumscribing circle for a given triangle.

Two graphs related to the Delaunay triangulation are the Euclidean minimum spanning tree and the Gabriel graph. Given N points $\{p_1, p_2, p_3, \ldots, p_n\}$ in the plane, the *Euclidean minimum spanning tree* is the tree of minimum total edge length whose vertices are the given points. The *Gabriel graph* is the graph such that there is an edge between p_i and p_j if and only if the circle with diameter $\|p_i, p_j\|$ centered midway between p_i and p_j does not contain any other point. These graph structures have the following hierarchy (Preparata and Shamos, 1985; Goodman and O'Rourke, 1997).

$$\mathrm{MST} \subseteq \mathrm{GG} \subseteq \mathrm{DT}$$

For surface reconstruction of 3D point sets, Hoppe et. al. (1992) propagate orientation information by traversing the MST embedded in the graph generated by the k-nearest neighbors of each point. The constrained Delaunay triangulation has been used for triangular mesh refinement over curved surfaces (Chew, 1993). Chew extends the circumcircle test to points embedded in a curved surface, thus generating a closed space curve embedded in the surface. If no points lie inside this loop then the triangle in question is a valid triangle.

The spatial proximity defined the Voronoi diagram would appear to be a natural basis on which to form a surface interpolating our particle system. The 2D Delaunay triangulation encodes information found in the Voronoi diagram as triangles, 2D geometric simplices, and thus can provide such a surface description. In addition the Delaunay triangulation has been shown to provide well shaped triangles and minimizes the roughness of surfaces defined as height fields. It also has an elegant description, the circumscribing circle test. For our purposes, the main drawback of 2D triangulation algorithms is that they are limited to reconstructing surfaces that uniquely map to the plane.

5.2.2 3D Structure

The Voronoi diagram and Delaunay graph naturally extend to higher dimensions. The dual of an D-dimensional Voronoi diagram is the D-dimensional Delaunay graph which partitions space into volumes bounded by D-dimensional simplices[4]. For N points, there are N regions in the Voronoi diagram. Unfortunately the number of items necessary to describe the Voronoi diagram (and the Delaunay triangulation) grows exponentially with the dimension (Preparata and Shamos, 1985, pg. 246). In the three dimensional case, the Voronoi diagram partitions space into N regions which are bounded by $O(N^2)$ edges and vertices in the worst case. Its dual, the three dimensional Delaunay graph is an $O(N^2)$ collection of tetrahedra tessellating the interior of the convex hull[5].

[4]A simplex in D dimensions is a spatial configuration determined by $D + 1$ points. A three dimensional simplex is a tetrahedron, a pyramid with four triangular faces.

[5]The convex hull is the smallest polyhedron such that all the points are contained within the volume. As a physical analogy, the convex hull can be thought of as applying shrink wrap plastic around a set of points. After the plastic shrinks to be a tight surface, the plastic surface is the convex hull.

While the 3D Delaunay triangulation defines a complete set of proximity relationships for a set of points, it does not provide a surface description, but rather a volumetric description of the data. We know that embedded in the Delaunay triangulation is a polyhedron which passes through all of the data points and is thus an interpolating surface. The problem now becomes: "How can we extract from this full set of proximity relations, a set of relationships which describe the surface?". If all points are on the convex hull, the convex hull is the surface. If not, one can remove simplexes from the set of tetrahedrons until all the points are on the boundary. Boissonnat (1984) solved this for surfaces of genus 0 (without holes) by providing a set of rules to iteratively remove simplices with a face on the boundary. The remaining tetrahedrons describe the shape's volume and the boundary faces define the surface. To my knowledge, this approach has not been extended to the general case of manifolds of arbitrary genus.

Alpha shapes were designed to allow scientists to explore the spatial structure of points sets, by extracting subsets of the Delaunay triangulation (Edelsbrunner and Mücke, 1994). Given a omnipresent ball with radius α, a polytope (a face or edge) of the triangulation is removed when the ball can enclose the polytope without enclosing any of the vertices. The resulting triangulation is the alpha shape. Surfaces can be reconstructed from point sets by extracting surfaces embedded in α-shapes (Guo, Menon and Willette, 1997), though there is a trade off between reconstructing over sparsely sampled areas and maintaining details in densely sampled areas. Unfortunately all of these techniques are based on first generating the 3D Delaunay triangulation, at a worst case time cost of $O(N^2)$ and expected cost of $O(N)$ (Dwyer, 1991).

5.2.3 Triangulation of Particles

Applying two dimensional surface reconstruction solutions suffice for small areas, but does not solve the larger three dimensional problem. Generating three dimensional proximity information does not simplify the problem, but transforms the problem into one of extracting a surface description from a volumetric description. Conceptually we would like the favorable qualities of a 2D triangulation to extend to triangulating our shape in 3D. One approach would be to locally compute a 2D triangulation by projecting the subsets of data onto a plane, triangulating the data points, and then projecting the triangulation back onto the original data set. Difficulties arise in determining what are appropriate subsets of points to consider and how to merge the resulting triangulations together. The real problem is that our data defines full three-dimensional shapes of unknown structure and not special cases which reduce to the two-dimensional domain. The N-dimensional Delaunay triangulation tests for inclusion or exclusion of N-dimensional simplices. In 2D it tests triangles. In 3D it tests tetrahedrons. We propose a test that takes the criterion of a two dimensional Delaunay triangulation and extends it into the three dimensional domain while preserving the two dimensional test.

In 2-D, a triangle is part of the Delaunay triangulation if no other vertices are within the circle circumscribing the triangle. To extend the circumscribing circle idea to 3-D, we check the smallest sphere circumscribing each triangle. This is also the 3D analogue of the 2D Gabriel graph. Given any three points, if no other points fall within the smallest circumscribing sphere, then these three points define a valid triangle. If another point lies within the sphere, the triangle is not part of the surface triangulation. Note that the smallest sphere circumscribing three points has embedded in it the smallest circle circumscribing the three points, which is the 2D Delaunay test. This circle is a great circle of the sphere.

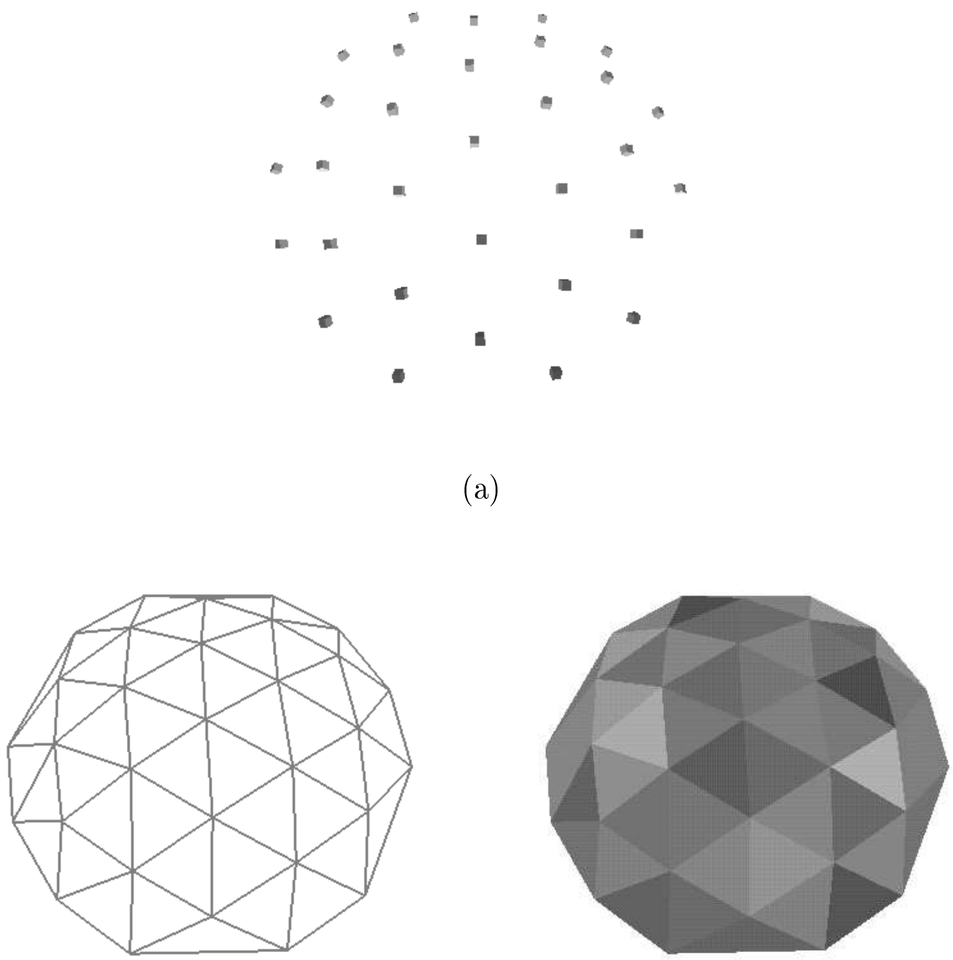
To avoid computing triangles in areas without particles, we limit the length of valid triangle edges (to 2 units of inter-particle spacing, by default). This also has the side effect of pruning the number of triplets to test from $O(N^3)$ to $O(N)$. The total cost of the triangulation is $O(N \log N)$; $O(N \log N)$ to search for all nearest neighbors and $O(N)$ to test for valid triangles. The reconstruction heuristic works well in practice when the surface is adequately sampled with respect to the curvature. To better visualize the resulting surface, Gouraud, Phong, or flat shading can be applied to each triangle. The results of using our triangulation algorithm are shown in Figure 5.3, where the original point set is shown along with the resulting triangulation displayed in wireframe and randomly colored filled triangles. Figure 8.17(e) is an example of a Gouraud shaded triangulation.

5.2.4 Undesirable Triangulations

The goal of the triangulation algorithm is to compute surface connectivity based on the spatial proximity of particles. However, the arrangement of the particles may not suggest a "reasonable" surface. For example, consider the four vertices of a regular tetrahedron (i.e. a tetrahedron made of equilateral triangles). From this set of vertices, there are four combinations of three vertices that one can choose. Each of these triplets corresponds to the face of the tetrahedron and each of these triplets will pass the minimum sphere test. Thus, given these four vertices, the triangulation algorithm will generate the faces of a regular tetrahedron. For this case, this is a reasonable surface to generate.

Now let us consider the case of two spherical arrangements of particles, say $\mathbf{V}_1$ and $\mathbf{V}_2$. Suppose that when $\mathbf{V}_1$ and $\mathbf{V}_2$ are far apart, the triangulation algorithm generates two triangulated spherical surfaces. Now further suppose $\mathbf{V}_1$ and $\mathbf{V}_2$ are moved closer together, such that two vertices from $\mathbf{V}_1$ and two vertices from $\mathbf{V}_2$ correspond to the vertices of a regular tetrahedron. It is possible that the triangulation algorithm will generate two spherical arrangements of triangles, plus the faces of a tetrahedron and hence connect the two sets of triangles together. This may be considered an "undesirable" triangulation of the points. We should note that generally such an arrangement of particles would not be a minimum energy configuration for an oriented particle system, unless there are external forces acting on the system.

As the two sets of particles are brought even closer together it is likely (depending on the full set of circumstances) that the two spherical arrangements of particles will merge, much like two soap bubbles will join with a wall between the two bubbles. For particles interacting under the influence of the Lennard-Jones, co-circularity, and co-planarity potentials, this would be a valid minimal energy configuration. Barring



(a)

(b) (c)

Figure 5.3: Triangulation of points

(a) A subset of points taken from a sphere of particles at equilibrium. (b) Triangulation displayed as wireframe. (c) Triangulation displayed as filled triangles of various colors.

large external forces, the co-normality potential could be used to prevent collections of particles with opposing normals from merging together. Such arrangements of particles are the collision of two spheres (as above) and the folding of a sheet of particles onto itself. With an appropriate choice of maximum triangle edge length, this may prevent the generation of triangles between the two sheets.

Since our triangulation algorithm imposes no constraints on the possible positions of the particles, we do not guarantee that the triangles generated will appear as a "reasonable" or "properly" connected surface. Given a set of samples from a known surface, it would be interesting to prove the limits of surface curvature and sampling density over which our triangulation algorithm can correctly reconstruct the surface. We leave this as an open problem.

We end this section on a philosophical note. The real question facing us is not, "Does the triangulation algorithm guarantee reasonable surfaces?", but rather the more general question, "What is reasonable surface connectivity for *arbitrary* sets of points?".

5.2.5 C^1 Continuity

For generating smooth surfaces from polygonal meshes, one can apply surface subdivision methods to replace the original polygon mesh with successively finer polygon meshes, which in the limit results in a curvature continuous smooth surface (Catmull and Clark, 1978; Peters and Reif, 1997) except at a number of extraordinary points (Doo and Sabin, 1978). Many subdivision schemes produce meshes which are a combination of quadrilaterals and convex polygons, with extraordinary points introduced for polygons which are not quadrilaterals and at vertices with edge valence not equal to four. Unfortunately, when applied to our triangulations, these approaches will introduce a high number of extraordinary points: approximately $2N$ for Peter and Reif's scheme and $3N$ for Catmull and Clark's scheme. Also, many of the polygons are slow to refine. Luckily there exist subdivision surfaces designed for triangulated surfaces of arbitrary topology. For example, Loop (1987) defines an approximating subdivision schemes specifically for triangular meshes. His method generates refined triangular meshes at each iteration and the extraordinary points are limited to the original mesh vertices. Zorin (1996) defines a subdivision scheme well suited for triangulations. In the limit, it interpolates a C^1 smooth surface between the original vertices .

For imaging purposes, increasing the resolution of the particle system can be as effective as fitting a C^1 surface. As the area of each triangle approaches the area of an image pixel, the visual difference between a C^0 surface and the corresponding C^1 surface becomes negligible. With the ability to render over 1 million anti-aliased texture mapped triangles per second on current low end graphics workstations to 80 million on high end systems[6], rendering large quantities of triangles is a feasible alternative.

[6]Statistics from SGI's Silicon Surf world wide web site (October 1997) discussing the SGI *O2* and *Onyx2 RealityMonster* systems.

For guaranteed continuity, there are spline based techniques for specifying C^1 and G^1 surfaces over polyhedra of arbitrary genus. Most techniques require the polyhedron to be a triangulated polyhedron, thus circumventing difficult cases. In the general case, the problem reduces to transforming a polyhedron into a triangulation. In our case this is not a problem, since our polygonization is already a triangulation. Guo and Menon (1996) and Bajaj and Ihm (1992) interpolate triangulations of arbitrary topology with implicit spline patches. Both assume normal vectors are defined at the vertices, and construct intermediate geometries as a precursor to constructing the surface patch. Loop (1994) presents an algorithm for approximating a triangular mesh of arbitrary topology with triangular surface patches that meet with G^1 continuity. The drawback of such techniques is that the generation of high order polynomial patches which are expensive to compute. A second, equally significant drawback in many applications (such as character animation) is the lack of control compared to hand crafted piecewise spline surfaces.

5.3 Summary

This chapter discussed methods of generating continuous surface descriptions from the discrete description provided by our particle system. We break the continuous surface description problem into two separate problems, one for volume particles and one for surface particles.

For particle systems of volume elements, we recommend an implicit surface approach to surface description. In this model, each particle is associated with a monotonically decreasing field in R^3 as a function of distance from the particle. This field corresponds to the volume of the particle. The summation of all particle fields defines another scalar field in R^3 which is a scalar field defined for the particle system, and iso-surface is defined by the locus of points in R^3 equal to a constant scalar "threshold" value. For a given point in space, testing the value of the field at that point against the threshold value computes whether the point is inside, outside, or on the surface. Polygonal approximations to the iso-surface can be constructed by sampling the field function on a regular grid and determining surface intersections between pairs of adjacent inside/outside points. Direct surface sampling techniques, such as ray-tracing, can also be used for rendering. For general field functions, iterative root solving techniques can be used to find the surface-ray intersections. For field functions defined by low order polynomials, analytic solutions can be found over discrete segments of the ray.

For particle systems of surface elements, we present an algorithm to construct triangulations over an even distribution of particle samples. Our algorithm is based on spatial proximity information as encoded in the Voronoi diagram and its dual, the Delaunay graph. However, neither 2D nor 3D Delaunay tests are directly applicable to our problem. In 2D the three point circumscribing circle test identifies valid triangles, and in 3D the four point circumscribing sphere test identifies valid tetrahedrons. Instead we wish to identify valid triangles in 3D. To do so we extend the essence of the 2D test to 3D. If no other points fall within the smallest sphere circumscribed by three points, these three points define a valid triangle of our surface. Note that in this test, the smallest circle circumscribing the three points (the 2D test) is a great circle of the sphere. To allow our surfaces to separate, we limit the lengths of triangle edges. This has the secondary advantage of reducing the number of particle triplets to test to be linear in the number of particles. For generating smooth surfaces from our triangulations, either subdivision surfaces or interpolating triangular spline patches can be used.